2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)
Bari, Italy. October 6-9, 2019

A Neural Word Embedding Approach to System Trace Reconstruction

Karuna Lakhani
Department of Computer Science
The University of British Columbia
BC, Canada
karuna25@live.com

Apurva Narayan
Department of Computer Science
The University of British Columbia
BC, Canada
apurva.narayan@ubc.ca

***Abstract*—Data generated by real-world systems specifically cyber-physical systems is full of noise, packet loss, and other imperfections. However, most intrusion detection, anomaly detection, monitoring, and mining algorithms and frameworks assume that data provided is of perfect quality. Therefore, these algorithms tend to perform extremely well in controlled lab environments but fail in the real-world.**

We propose a method for accurately restoring discrete temporal or sequential system traces affected by data loss, using Word2vec's Continuous Bag of Words (CBOW) model. The model works by learning to predict the next event in a sequence of events, the model feeds its output back into it for subsequent future predictions. Such a method can reconstruct even long sequence of missing events, and help validate and improve data quality for noisy data. The restored traces are very close to the real-data and can be used by algorithms depending on real-data for system analysis. We demonstrate our method by reconstructing traces from QNX real-time operating system consisting of long sequences of discrete events. We show that given even small parts of a QNX trace, our CBOW model can predict future events with an accuracy of almost 90% outperforming the Markov Model benchmark.

I. Introduction

In context of software systems, it is often observed that their behavior varies significantly at times in real-world as compared to the controlled development environment. This becomes a threat when the software is deployed on a safety-critical system such as an aircraft, pacemaker, car etc. In such systems, the slightest malfunction of the system software can lead to catastrophic outcomes. Therefore, it becomes extremely important for our algorithms and softwares, to operate as intended in all environments. The underlying reason for different behavior in different environments is presence of imperfect data. In a lab setting, the algorithms are developed using clean, pre-processed datasets that is challenging to get in the real-world datasets. In real-world data there are imperfections such as loss of events, corrupt data etc. This results in failure of the algorithms to perform in practice.

Fixing of real-data is considered a daunting task. However, if appropriate historical information and context is available the data sets can be fixed at a minimal computational expense. Various machine learning techniques can be used to learn the contextual information in the data sets and subsequently be used for recovering lost data that is of interest.

Many real-world applications generate discrete event data during their operation that are called event traces or log files. For instance, the QNX real-time operating system (RTOS) is used in many safety-critical systems, such as medical devices, nuclear monitoring systems, vehicles, and so forth. The QNX RTOS has a very advanced logging facility, *tracelogger*. *Tracelogger* facilitates detailed tracing of the kernel and user process activity on any system. More specifically, it can log interrupt activity, various states of processes and threads, communication within the system, kernel calls, custom user events, and much more. The logged events give a detailed view into the behavior of the system, but at times the logs have missing data due to issues such a buffer overflow that makes them difficult to make use of by developers and system designers for analyzing system behavior. These traces are thus a perfect resource for developing a model for recovering information and validating the performance of the model.

Process mining relates to extracting knowledge from events log to improve real time processes. The main idea is to find out some information from event logs which are induced by systems. However, missing and corrupt data makes these datasets unusable. Process mining techniques are applicable in many different fields such as Customer Relation Management (CRM), workflow management, product development, healthcare, Business to Business (B2B) and so on [1]. The goal of process mining is to draw knowledgeable information about real time processes. The data induced from real time systems is used to repair process models. Process mining is the state of art which fills the gap between process analysis and data oriented analysis.

Neural networks have been used extensively to represent words as continuous vectors. Learning distributed representations of words that try to minimize computational complexity is an important aspect since most of the complexity is caused by the non-linear hidden layer in the model. Word2vec [2] is a family of techniques for encoding words as relatively low-dimensional vectors that capture interesting semantic information. That is, words that are synonyms are likely to have vectors that are similar. Another really neat aspect of this encoding is that linear transformations of these vectors can expose semantic information like analogies: for example, given a model trained on a system trace from an automotive, adding the vectors for `LEFT_INDICATOR`, `STEERING_TURN_LEFT` and `LEFT_INDICATOR` results in a vector very close to that for `STEERING_ANGLE_NON-ZERO`.

978-1-7281-4569-3/19/$31.00 ©2019 IEEE

Word2Vec [2] has been used extensively for work with NLP and event based data [3], data compression [4], and many other applications. However, the use of Word2Vec type of models has not been explored in restoration of lossy traces. We propose a method for using Word2Vec as an underlying framework to accurately reconstruct discrete temporal traces from QNX Real-time operating system. These reconstructed pseudo-perfect traces can be then used by algorithms that rely on the availability of perfect data.

The model works by learning the semantic relationship between events in the traces using Word2Vec approach as explained later. The model then attempts to predict the next event in the sequence given current and a subset of historical events. It conditionally uses its own output as an input to the model to predict subsequent events. This approach enables the recovery of large chunks of traces. These reconstructed traces can be used by various algorithms that depend on perfect data such as those designed for run-time monitoring or specification mining [5].

The rest of the paper is divided as follows: Section II will discuss related work in context of lossy data recovery, Section III will explain our approach, Section IV explains the experimental setup, the data used from the QNX operating system, and Section V will present results and conclusion.

II. Related Works

The application of novel machine learning and deep learning techniques has been increasing rapidly leading to the design and development of state-of-the-art solutions to numerous real-world problems in the domain of computer vision [6], robotics [7], machine translation [8], etc. In context of sequence prediction, numerous techniques have been developed to solve the problems in the domain of Natural Language Processing (NLP) [9], stock market prediction [10], music generation [11], and many others.

There have been numerous studies that use deep learning and natural language processing that talk about anomaly detection [12, 13, 14], and intrusion detection [15, 16, 17], very few focus of the issue of restoring lossy or noisy data that is critical for analyzing the behavior of the system to be correct. Text representation learning has been extensively studied. Popular representations range from the simplest Bag of Words (BoW) and its term-frequency based variants [18], language model based methods [19, 2, 20], topic models [21, 22], and distributed vector representations [23, 24, 25] or LSTM based approaches [26, 27].

Often, missing data are considered unusable and is removed completely in pre-processing to avoid having to address it. When missing data are addressed, missing values are typically replaced with global or class means [28, 29] or neighboring values [30, 31]. These approaches are superficial and do not consider the specific sequence of events. In [32] they discuss this problem and propose a novel LSTM-based approach they call an "Iterative Imputing Network" for restoring missing sensor data in time-series. While their work is on restoring continuous, multivariate data, we draw inspiration from it and propose a method using CBOW to restore discrete, sequential, univariate data.

The key contributions of this paper include:

- A novel approach to identify implicit relationship between processes in complex systems from event traces;
- A framework for recovering lossy event traces in complex systems;

III. Framework for Trace Modelling

A. Word2Vec: Continuous Bag of Words (CBOW)

In this paper, we employ Continuous Bag of Words (CBOW) Word2Vec [2] model to learn event features. Word2vec was developed by Tomas Mikolov in 2013 at Google, is a two layered neural network model which is trained to learn word embedding (embeddings are context based low dimensional representation of words). Word2Vec comprises of two different learning models, Continuous Bag of Words (CBOW) and Skip-gram [33, 34]. CBOW predicts the center word given its context, whereas Skip-gram is opposite of CBOW. Skip-gram predicts the context word given a center word. CBOW predicts the word given its context, but Skip-gram predicts the context given a word. After training, Word2vec model generates the word vectors from the training corpus and learns the vector representations of each word. Word2vec also calculates the similarity among the words.

CBOW Word2vec model: The architecture of a CBOW model is represented in Figure 1. Let us assume that the current

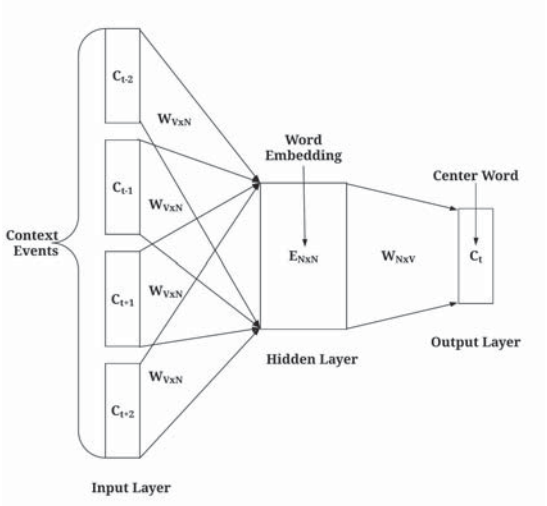

Fig. 1: Architecture of CBOW Word2Vec model

word in a sentence C is c_t. The input to the model could be the context words of current word c_{t-2},c_{t-1},c_{t+1},c_{t+2}. The model tries to predicts the center words c_t from the input context words. The training objective is thus defined as:

$$\mathrm{J}_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(c_t : | : c_{t-n}, \cdots, c_{t-1}, c_{t+1}, \cdots, c_{t+n}) \quad (1)$$

where n is the window of words around the center word c_t at each time step t. The term $\mathbb{P}(c_t \mid C_o)$ where c_t

represents the center word at time step t and C_o are the context words for center word. This probability is calculated by a softmax function. However, calculating gradient of this term is computationally expensive. To overcome this issue we use the alternate hierarchical softmax function to solve the above problem is proposed in [35]. The Word2Vec model performs the following four tasks:

- Build the corpus vocabulary;
- Build a CBOW (context, target) generator;
- Build the CBOW model architecture;
- Train the Model;
- Get Word Embeddings

We used `gensim` python module to build our deep learning architecture for CBOW model. The input to this model are a random length corpus generated from context words. The activation function for output layer is softmax which predicts our target word. The output word is matched with actual context word to compute loss and then perform back propagation with each epoch to learn the word embedding. Training the model takes considerable amount of time due to size of corpus. After the model is trained, similar words will have almost similar weights and will be placed closer to each other in vector space. We can get embedding vectors easily using `keyedvectors` module is `gensim` library. The following Figure 1 describes the network architecture.

In the next subsection we present the overall workflow of our trace recovery framework..

B. Workflow

Figure 2 provides a high level overview of the trace restoration framework. We divide our workflow into two parts, training and implementation

During the training phase, we initially train the model using perfect traces to learn the underlying relationships among the events in the trace. We generate random length sequences from the extracted events that act as sentences for training the model. These sentences are provided into the constructor of a `gensim` Word2Vec() instance. There are a few hyperparameters that can be configured to the constructor such as:

- $size$: The number of dimensions of the word embedding.
- $window$: The maximum interval between a center word and neighboring words.
- $workers$: The number of threads to use while training.
- sg: The training algorithm, either skip gram (1) or CBOW (0).
- min_count: Minimum count of occurrence of words to be considered.

The Word2Vec CBOW model is trained using the backpropagation algorithm. Lastly, a trained model can then be saved to file.

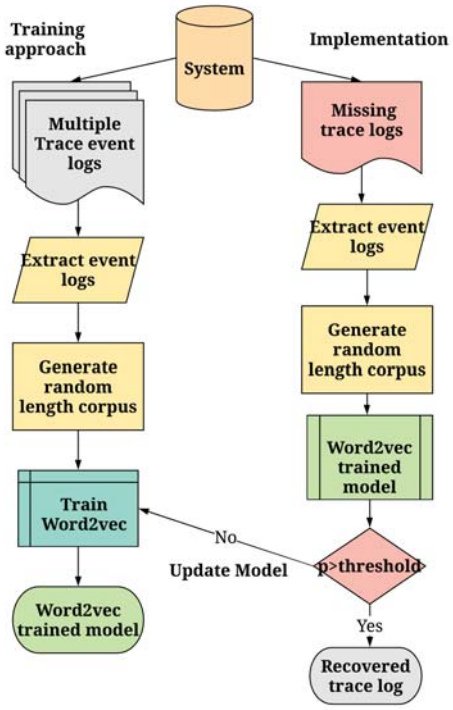

Fig. 2: Workflow for Trace Modelling and Restoration

During the implementation phase, we use lossy traces with missing events and try to predict the missing event using trained model. Firstly, We extract events from this lossy traces and generate context events to feed into the model. The length of the context events is a hyperparameter which can be set depending upon the data set and purpose of the model. Generally, the length is twice the window size set during the training phase. The output from the trained model is the probability distribution of the center event given context events. To accept the predicted output of the model, we define certain threshold p. This threshold helps to decide whether the predicted event is to be selected.

Algorithm 1 Pseudo Code for training Word2Vec model

```
Input: A set of perfect traces t
Output: Word embedding
Extract events e from traces t
for all events e do
    Convert event traces into strings
end for
while w ≤ len(e) do
    k ⇐ random(λ1, λ2)
    Generate sentences c from e
end while
Generate vocabulary v from c
Train CBOW Word2Vec model with window size s
```

Algorithm 1 presents the pseudo-code for training Word2Vec using complete traces.

For training the model, we use perfect traces of the system

called as t. The first step is to extract events from traces. We call this events as e. We then generate random length sentences from this traces. We assume λ_1 to be a minimum length and λ_2 to be a maximum length of sentences to be formed from events e. λ_1 and λ_2 are hyperparameters which can vary depending on the purpose of model training and the dataset. After this process, the object c contains the tokenized sentences of all the events in the traces. From c, we then generate a vocabulary of unique events v. Training CBOW using `gensim` is straightforward. We select a window size of s which slides through the corpus during training. Each event is a prediction problem. With the help of context events, the model tries to predict the center event. The predicted result determines whether to improve the current event vector. The model is trained using backpropagation.

Algorithm 2 Pseudo code for recovering lossy traces using trained Word2Vec

```
Input: A set of traces with events missing at points tm
Output: Recovered missing event em at point
Extract events m from lossy traces tm
for all events in m do
    Convert trace events into strings
end for
Load the trained model
Generate list of context events cm with size s
Predict the missing event em
Check the accuracy of model α
if α ≤ λ then
    Retrain the model with complete traces
else
    Do nothing
end if
```

Algorithm 2 presents the pseudo-code for recovering lossy events from traces using trained Word2Vec. We use lossy traces tm with missing events em. Using pre-trained model missing event em is predicted. To recover a lossy trace, we initially extract events from the trace. We name this list of extracted events as m. Next, we generate a list of context event pairs cm with window size w. w is usually twice the window size used for training the model. We load the pre-trained model and provide cm to it. The model then predicts a list of events along with its probability. The event which had the highest probability amongst all the other predicted events is then choosen as predicted missing event em. In order to account for constant changes in the system, some level of retraining is necessary. So, we check for the accuracy of the model $alpha$ rate of correctly predicted events. If the accuracy of the model is below the required accuracy λ, we retrain the model using complete traces. Thereby we make sure the accuracy of the model is maintained.

In the next subsection we will discuss about the data used to demonstrate the strength of our model.

C. Data

The data is obtained from a commercially available gyro-stabilized Mikrokopter hexacopter equipped with 6 electric motors and a 6200 mAh lithium polymer battery [36]. The vehicle is capable of 10 minute flights when loaded with the payload described below, and is constructed with a custom autopilot which is a secure, tested, embedded platform with integrated IMU, GPS and pressure sensors that is capable of fully autonomous position control operation.

The system is also able to operate in a traditional RC flight mode, where a pilot can operate the vehicle by providing roll, pitch and yaw rate commands as well as thrust commands to the vehicle. This mode of operation does still include gyro stabilization of the vehicle based on a separate set of gyroscopes connected to the COTS receiver, making flight significantly easier by adding sufficient damping to the system and nominally stabilizing the vehicle. Switching between autonomous and RC flight mode is accomplished by a separate hardware switch, triggered by the pilot over the 2.4 GHz RC channel.

The software runs on top of QNX Neutrino 6.4, and the target board is a Beagleboard. The data was collected from the system using the QNX tracelogger facility using the configuration to capture all classes and using the wide option to collect the maximum amount of information. The time resolution for time-stamps is in nanoseconds, traces are in general short runs covering a few seconds of operation.

We collected a total of 40 traces during the operation of the hexacopter. The hexacopter performed various maneuvers during the flight. Of the 40 traces, 32 were used for training and validation, while 8 were held out for testing. The traces used for training were examined to compose a dictionary of possible event IDs. A total of 209 different event IDs were found in the training set. We then one-hot encoded all traces using this dictionary, replacing the one-dimensional event ID with a 36-dimensional vector of indicator variables where 209 elements have the value 0 and 1 element the value 1.

Using QNX tracelogger traces, we built our corpus vocabulary by generating unique events from the traces. Because Word2Vec trains itself based on this vocabulary set, we need to generate vocabulary from all the available 40 traces. Our Vocabulary consisted 209 words. For implementation of CBOW, we need to generate pairs which consist of a center word and surrounding context words. For this paper, a center word is of length 1 and context words are of length $2 \times s$ where s is a hyper-parameter usually equal to window size.

D. Training

The CBOW was trained using the following procedure:

1) Select 2 traces from set (1 for training, 1 for validation)
2) Train the model on these 2 traces for 10 epochs with a learning rate of 0.025, window size of 1 and a softmax loss function
3) Train the model on these 2 traces for an additional 15 epochs with learning rate decay of /1.1 and a softmax loss function
4) Reset learning rate, preserve weights, repeat from step 1 selecting 2 new random traces

Dropout for regularization and randomized input order prevent overfitting, so this procedure should be repeated until accuracy

reaches a plateau since early termination due to detection of overfitting is unlikely to occur. Alternatively, the procedure can be repeated a fixed number of times if so desired.

E. Benchmark

In order to understand what the results mean, it is important to have a benchmark. We create a benchmark using a fast Markov Model that could be described as a history search, or as a conditional probability method. It has been shown that in situations where data or computational power are limited, Hidden Markov Models can match the performance of state-of-the-art LSTMs [37] or other deep learning models. As a result, the Markov Model will act as a benchmark that the CBOW can be compared against.

IV. EXPERIMENTS

The basic idea of this approach is that the next event is highly correlated with the preceding events in the sequence. The strength of this correlation degrades as the algorithm look further in history. As a consequence, we can claim that the system is a Markov process of order n, where n previous events affect the upcoming state of the system. Such sequences of n events are referred to as "n-grams", and they are commonly analyzed in NLP [38, 39, 40]. To simulate similar behavior we use the approach proposed by [41] where they build a model of the system by selecting n, and creating a separate state for each possible combination of the d unique messages. They initialize all possible states at once which may quickly become too memory-intensive for large n or d. To mitigate this issue, they iteratively fill a dictionary D by traversing the training data based on a algorithm presented in [41].

In lossy data, multiple consecutive events may be lost. This makes predicting multiple sequences ahead in time an important and challenging problem. For predicting next events from a given data we initially increased the number of context events to be fed into the model. We start with one event at a time and predict one future event. The first order Markov model [42] was trained using entire data set and next event was predicted. The results were compared with CBOW Word2Vec model. We calculated accuracy of the model based on how many set of events were correctly predicted from given set of sequence. Accuracy of prediction is calculated using equation below,

$$\alpha = \frac{c_p}{c_p + c_n} \tag{2}$$

where α is the accuracy of the proportion of events that are correctly predicted, c_p is the number of events correctly predicted and c_n is number of events which were incorrectly predicted. The results of the experiment are illustrated in Table I below where n is the number of forward steps to be predicted by model. In a sequence of events, if $n = 1$, we are trying to find the accuracy of model to predict one event at time. Similarly, if $n = 5$ then, our model will predict 5 events at a time. For this, we feed the output of previous stage to CBOW and predict the next word for 5 times. We experimented on set of 20 events and found that as the number of predicting future events increases, the accuracy of CBOW degrades.

n	CBOW accuracy	Markov Model accuracy
1	0.90	0.15
3	0.80	0.05
5	0.60	0.05

TABLE I: The table presents comparison of accuracy in % for CBOW and Markov Model

	Predicted Event	True Event
1	7	7
2	8	8
3	9	9
4	10	10
5	8	8
6	11	11
7	10	10
8	12	12
9	96	96
10	13	13

TABLE II: The table compares the predicted vs true events in the traces

In Table II, we present the results for the case when CBOW model has to predict one event at a time. We take set of 20 events from the test data and try to predict the next event, one at a time. We performed this experiment for 2 traces. We can see the model was able to predict all the events in sequence correctly. This was not the case with our base case of Markov model. In Table III, we present the results for the case where

	Predicted Event	True Event
1	112	12
2	11	14
3	3	3
4	122	122
5	130	93
6	109	130
7	3	3
8	122	122
9	130	130
10	128	128

TABLE III: The table shows the recovery of events after few misclassified events

model tries to stabilize after few events and again predicts correctly. This is because infrequent words in CBOW model do not appear often during training model. It reflects the robust behavior of the model.

Figure 3 presents a accuracy of predicted next event. To check how prominently the model can predict, we conduct an experiment where we start predicting next event by giving feedback to the model. We initially consider a random event lets say e. We input the model this event e and predict the next event ep. The predicted event ep is given as an input to the model in the succeeding step and again tries to predict the subsequent event. We repeat the experiment for predicting events with a step size of one, three and five. Step size determines how many events to predict at a given point. Let's say we only want to predict just next event at a point our step size will be one and if we want to predict the next three events at a point, then our step size will be three. The results show that the model performance is better by predicting just

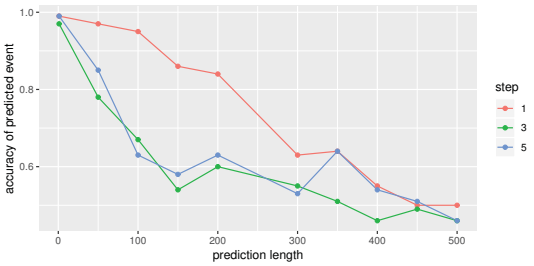

Fig. 3: Feedback accuracy of CBOW

the next event. The accuracy of the predicting up to 200 events is above 80% for a step size of one whereas it declines after the prediction of 50 events for a step size of three and five. Our experiment results confirms that the proposed method is significantly better than Markov Model in terms of prediction accuracy and recovering lossy traces. The prediction accuracy of predicting one event is better then predicting multiple events at a given point. That's because it is memory less model and depends only on context of words around the center word. The hyperparameters during train phase play a key role in deciding the model capability. The CBOW Word2Vec based lossy trace recovery model is sufficiently robust and significant to be used as data pre-processor algorithm, where recovering lossy or missing data is critical to understand the behavior of the missing data. In the following section we will present our results and conclusion with some context for future work.

V. Conclusions and Future Work

Availability of clean data is critical for ensuring correct and reliable behavior of not only various software systems but also training of machine learning and deep learning algorithms. Unfortunately, availability of clean and perfect data in the real-world is a myth. The data is quite often subjected to noise, loss, corruption, and other imperfections. We have developed and presented a CBOW based approach for recovering lost or noisy data in cases where the data is in the form of discrete event traces. This approach, if trained sufficiently well will enable the system operators to feed back the output to continuously keep predicting in the future. One of the main advantages of our approach is that it does not need to know anything about the underlying system where the systems have temporal dependencies.

Our CBOW-based method is nonetheless an effective standalone solution for recovering lost discrete data in the field. It can be used to pre-process data intended to be used with any other algorithm in order to improve end-to-end performance.

References

[1] W. M. P. van der Aalst and A. J. M. M. Weijters, "Process mining: A research agenda," *Comput. Ind.*, vol. 53, no. 3, pp. 231–244, Apr. 2004.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[4] K. Filippova, E. Alfonseca, C. A. Colmenares, L. Kaiser, and O. Vinyals, "Sentence compression by deletion with lstms," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 360–368.

[5] A. Narayan, G. Cutulenco, Y. Joshi, and S. Fischmeister, "Mining timed regular specifications from system traces," *ACM Transactions on Embedded Computing Systems*, vol. 17, 2018.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in neural information processing systems*, 2017, pp. 1087–1098.

[8] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[9] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[10] M. R. Vargas, B. S. L. P. de Lima, and A. G. Evsukoff, "Deep learning for stock market prediction from financial news articles," in *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, June 2017, pp. 60–65.

[11] M. C. MOZER, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing," *Connection Science*, vol. 6, no. 2-3, pp. 247–280, 1994.

[12] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct 2016, pp. 130–139.

[13] T. Ergen, A. Hassan Mirza, and S. S. Kozat, "Unsupervised and Semi-supervised Anomaly Detection with LSTM Neural Networks," *ArXiv e-prints*, Oct. 2017.

[14] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*. Presses universitaires de Louvain, 2015, p. 89.

[15] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26.

[16] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on*. IEEE, 2016, pp. 258–263.

[17] M.-J. Kang and J.-W. Kang, "Intrusion detection system

using deep neural network for in-vehicle network security," *PloS one*, vol. 11, no. 6, p. e0155781, 2016.
[18] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, Aug. 1988.
[19] W. B. Croft and J. Lafferty, *Language modeling for information retrieval*. Springer Science & Business Media, 2013, vol. 13.
[20] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
[21] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.
[22] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
[23] G. Mesnil, T. Mikolov, M. Ranzato, and Y. Bengio, "Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews," *arXiv preprint arXiv:1412.5335*, 2014.
[24] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, 2014, pp. 1188–1196.
[25] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.
[26] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.
[27] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in neural information processing systems*, 2015, pp. 3079–3087.
[28] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, and K. G. Moons, "A gentle introduction to imputation of missing values," *Journal of clinical epidemiology*, vol. 59, no. 10, pp. 1087–1091, 2006.
[29] T. Schneider, "Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values," *Journal of climate*, vol. 14, no. 5, pp. 853–871, 2001.
[30] G. Kalton, "Compensating for missing survey data." 1983.
[31] P. Royston *et al.*, "Multiple imputation of missing values," *Stata journal*, vol. 4, no. 3, pp. 227–41, 2004.
[32] J. Zhou and Z. Huang, "Recover missing sensor data with iterative imputing network," *CoRR*, vol. abs/1711.07878, 2017.
[33] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 3111–3119.
[34] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations." in *HLT-NAACL*, 2013, pp. 746–751.
[35] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model." in *Aistats*, vol. 5. Citeseer, 2005, pp. 246–252.
[36] Rizwan, Yassir, "Towards high speed aerial tracking of agile targets," 2012.
[37] M. Panzner and P. Cimiano, "Comparing hidden markov models and long short term memory neural networks for learning action representations," in *International Workshop on Machine Learning, Optimization and Big Data*. Springer, 2016, pp. 94–105.
[38] M. Schonlau, N. Guenther, and I. Sucholutsky, "Text mining with n-gram variables," *Stata Journal*, vol. 17, no. 4, pp. 866–881, 2017.
[39] Y. Lin, J.-B. Michel, E. L. Aiden, J. Orwant, W. Brockman, and S. Petrov, "Syntactic annotations for the google books ngram corpus," in *Proceedings of the ACL 2012 system demonstrations*. Association for Computational Linguistics, 2012, pp. 169–174.
[40] G. W. Lesher, B. J. Moulton, D. J. Higginbotham *et al.*, "Effects of ngram order and training text size on word prediction," in *Proceedings of the RESNA'99 Annual Conference*. Citeseer, 1999, pp. 52–54.
[41] M. S. S. F. Ilia Sucholutsky, Apurva Narayan, "Deep learning for system trace restoration," in *2019 International Joint Conference on Neural Networks*. IEEE, 2019.
[42] P. A. Gagniuc, *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.